Towards Better Integration of Surrogate Models and Optimizers *

Tinkle Chugh, Alma Rahat, Vanessa Volz, and Martin Zaefferer

Abstract Surrogate-Assisted Evolutionary Algorithms (SAEAs) have been proven to be very effective in solving (synthetic and real-world) computationally expensive optimization problems with a limited number of function evaluations. The two main components of SAEAs are: the surrogate model and the evolutionary optimizer, both of which use parameters to control their respective behavior. These parameters are likely to interact closely, and hence the exploitation of any such relationships may lead to the design of an enhanced SAEA. In this chapter, as a first step, we focus on Kriging and the Efficient Global Optimization (EGO) framework. We discuss potentially profitable ways of a better integration of model and optimizer. Furthermore, we investigate in depth how different parameters of the model and the optimizer impact optimization results. In particular, we determine whether there are any interactions between these parameters, and how the problem characteristics impact optimization results. In the experimental study, we use the popular Black-Box Optimization Benchmarking (BBOB) testbed. Interestingly, the analysis finds no evidence for significant interactions between model and optimizer parameters, but independently their performance has a significant interaction with the objective function. Based on our results, we make recommendations on how best to configure EGO.

Tinkle Chugh
University of Jyvaskyla, Faculty of Information Technology, FI-40014 University of Jyvaskyla, Finland,
Department of Computer Science, University of Exeter, UK
e-mail: tinkle.chugh@gmail.com

Alma Rahat
School of Computing, Electronics and Mathematics, University of Plymouth, Plymouth, UK
e-mail: alma.rahat@plymouth.ac.uk

Vanessa Volz
School of Electronic Engineering and Computer Science, Queen Mary University of London, London, UK
e-mail: v.volz@qmul.ac.uk

Martin Zaefferer
Faculty of Computer Science and Engineering Science, TH Köln, Steinmüllerallee 1 51643 Gummersbach, Germany e-mail: martin.zaefferer@th-koeln.de

* All authors contributed equally to this work

1 Introduction

Many real-world optimization problems depend on computationally expensive black-box simulations or experiments. Examples are the design and optimization of aircraft [34] or chemical reactors [31], which may require significant computational or financial resources for each evaluation. Therefore, decision makers and optimization analysts usually seek a good solution using a minimal amount of expensive evaluations.

In the last few decades, evolutionary algorithms (EAs) became popular for both single- and multi-objective optimization problems because of their several advantages in this context. For instance, they usually do not assume any convexity and differentiability of the objective or constraint functions [10, 12] and are thus well suited to solve black-box problems. However, because of their exploratory nature, EAs often need a considerable number of function evaluations to approximate optimal solutions. For problems with computationally expensive functions, surrogate-assisted evolutionary algorithms (SAEAs) aim to alleviate this weakness by replacing some evaluations with estimates from surrogate models.There are two main approaches to the integration of surrogate and optimizer: (1) The SAEA alternates between improving the surrogate model and improving the estimate of the optimum (via an optimizer operating on an acquisition function) and (2) certain parts of the SAEA (e.g. selection in evolutionary algorithms) are enhanced by the knowledge obtained through the surrogate model. In this study, we will be focusing on the first approach as visualized in Fig. 1. For more details about the second approach, see [9, 22].

The reason for choosing the first approach is that it is comparatively more modular, and hence more straightforward to explain through experiments. Henceforth, we refer to the first approach as SAEAs within the context of this chapter.

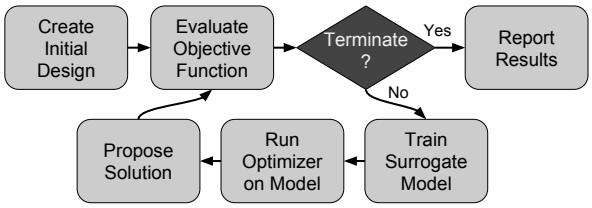

Fig. 1 The investigated SAEA process. The expensive objective function is optimized by iteratively optimizing a surrogate model based infill criterion. The optimizer is used to select solutions for evaluation with the expensive objective function. The model is then updated accordingly and the next iteration starts. The process terminates after the budget of expensive function evaluations is exhausted.

As the first step, an initial set of samples is generated (e.g. using a design of experiment technique [30]) as indicated in the top left corner of the figure. These samples are evaluated with the expensive objective function and the derived data

is then used to train the surrogate model. An optimizer (e.g. an evolutionary algorithm) is applied to find sample(s) for updating the surrogate, based on a so-called infill criterion (or acquisition function or updating criterion). The selected sample(s) are evaluated with the expensive evaluation(s) and combined with the previously evaluated samples. This process is repeated until a termination criterion such as a maximum number of expensive evaluations is met. The solution corresponding to the minimum objective function value is used as the final solution among all the evaluated ones.

In the literature, several SAEAs have been proposed for use cases with small evaluation budgets. These algorithms can mainly be distinguished by the infill criterion they use, i.e. their strategy for selecting new sample(s) to update the surrogate model. Different strategies have been proposed in the literature, e.g., expected improvement [23], lower confidence bound [35], and probability of improvement [11]. For more details, e.g., advantages, limitations and other properties, see [9, 22].

In theory, any optimizer (or configuration of an optimizer) can be coupled with any type of surrogate model. However, there is a lack of extensive studies or detailed guidelines in the literature on co-configuring the optimizer and the modeling approaches for improved performance. For instance, different modeling techniques, e.g. neural networks, Kriging, and support vector regression, have their own advantages and limitations. In addition, evolutionary algorithms have certain parameters which can affect the performance of the SAEA. Therefore, an efficient integration of two elements can be useful in increasing the performance of the algorithm. In the next subsection, we present our hypothesis to incorporate two major elements, i.e. surrogate and EA when developing and applying a SAEA.

1.1 Hypothesis

As mentioned above, numerous SAEAs have been proposed in the literature, incorporating surrogates and evolutionary algorithms in different ways. Most of these approaches ignore the potential benefits of a proper integration of the two parts. For example, harmonizing the correlation assumptions of the surrogate model and the variation operators could speed up the optimization process. Another idea is to choose and improve the surrogate model depending on what information is actually used by the EA.

Moreover, EAs generally do not make assumptions about the properties of the fitness landscape, but surrogate models often do. Additionally, the properties of the model output are known and can be considered when selecting or configuring the optimization algorithm. Practitioners and algorithm developers should be concerned with choosing the algorithm and the type of variation operator that works best with certain model assumptions, to avoid deteriorating optimization performance. Other important issues to be considered are the allocation of a computational budget to the model, and the optimizer and the balancing of the trade-off between exploration and

exploitation. In this chapter, we investigate the interaction between the EA and the surrogate model, motivated by the following hypothesis.

Hypothesis. The two main components of SAEAs (optimizer & model) closely interact with each other, thus influencing the overall algorithm performance.

That means, configuring the components separately may be detrimental for the overall optimization performance. For example, the chosen EA could inform the decision about the infill criterion and the error metrics used to train the surrogate model. There is also a possibility of dynamic mutual adaptation of model and EA as information is gathered during the optimization process and the problem starts to change from a black-box to a gray-box. For instance, both EAs as well as the surrogate models define some concept of neighborhood. In EAs, the neighborhood of a solution is determined by a variation operator. In the surrogate model, the neighborhood of a solution may be represented by a similarity measure (e.g. the correlation function in a Kriging model). Aligning both concepts of neighborhood may improve performance: the similarity measure may be used to derive a variation operator, and vice versa.

As a transparent example, consider a modeling approach that is prone to produce piecewise constant, and non-smooth surrogate models of the fitness landscape, e.g. tree-based models like random forest as used in [21]. Some simple hill climbing or gradient-based optimization algorithm may easily fail for such a model. Even certain design choices of more complex algorithms, e.g. the local search procedure in a memetic EA [33], may provide poor results in that case. Of course, such an issue would be rather obvious. But it nicely showcases how the choice or configuration of surrogate model and optimizer can interact. In the next subsection, we present our proposal and methodology for investigating the effect of different elements in the surrogate and the EA.

1.2 Proposal and methodology

In this work, we investigate the following research questions to study the interactions between the surrogate model and the EA-based optimizer:

- How does the choice of surrogate model affect the optimization process?
- How does the choice of optimization algorithm affect the optimization process?
- Are there any interactions between both elements?
- How do they relate to characteristics of the optimization problem?

Thus, we aim to lay the groundwork for a more efficient fusion of optimization algorithm and surrogate model.

To enable a transparent and clearly arranged analysis, this study considers only single-objective optimization problems. In order to focus the analysis, we concentrate on Kriging models. The Kriging or Gaussian process model [14] is one of the most common surrogate techniques, mainly because of its ability to estimate

the prediction error/uncertainty measure. To ensure comparability, we also consider only efficient global optimization (EGO) [23] described in Sect. 2 as an integration approach, i.e. expected improvement-based concurrent improvement of the surrogate and optimization the problem. As a testbed for our experiments, we use BBOB [17] and its single-objective problem suite [18] intended to represent a wide range of problems. We focus on the following aspects by conducting several experiments and show the effect of different algorithm configurations. In detail, we investigate:

- Effects of surrogate model configuration on the optimization process:
  - Kernel properties and relationship with test functions.
  - Initial sample size.
- Effects of optimizer configuration on the optimization process:
  - Choice of variation operators and relationship with the infill criterion landscape induced by kernel functions.
- Interactions and observed patterns with respect to characteristics of the fitness landscape.

The rest of the chapter is organized as follows. In the next section, we present the working methodology of Kriging and a brief description of the EGO algorithm considered in this study. In Sect. 3, we elaborate on integrating surrogate models and optimizers, and present relevant approaches from the literature. In Sect. 4, we describe the experiments conducted to address the hypothesis and discuss the results. Finally, we conclude and suggest future research directions in Sect. 5.

2 Efficient Global Optimization (EGO)

A very effective optimization framework for expensive single-objective problems is Efficient Global Optimization (EGO). It is widely used in the literature (for more details, see a recent review [39] and references therein).

In essence, EGO is a model-based sequential search strategy that samples the design space at likely locations of the global optimum as indicated by a surrogate, i.e. a subclass of the surrogate-assisted approaches depicted in Fig. 1. It starts with a space filling design, usually a Latin Hypercube Design (LHD) [30], of the decision space. The initial design samples are evaluated with the expensive objective function. With the observed mapping from decision space to objective space, a surrogate (regression) model based on Kriging is built. It should be noted that any model capable of producing a predictive distribution may be used within the EGO framework, but here we choose to use Kriging.

Once trained, the model generates a global posterior predictive (normal) distribution. As such, querying the model at any potential position in the decision space may indicate how likely it is to achieve an improvement over the best function value observed so far and how large this improvement may be. This particular measure

of utility (or infill criterion) is the expected improvement, and the benefit of using Kriging is that it permits the exact computation of the expected improvement. Furthermore, the expected improvement has monotonicity properties: it is inversely proportional to the predicted mean (with fixed uncertainty), and directly proportional to the uncertainty in prediction (with fixed mean prediction). It therefore strikes a balance between exploration and exploitation, and thus it is the most commonly used infill criterion in EGO.

Consequently, an obvious strategy to select the next solution to be evaluated by the expensive objective function is to select the solution that maximizes the infill criterion. This newly evaluated solution is then added to the current database and the Kriging model is retrained. This process is repeated until the budget of expensive function evaluations is exhausted.

Without loss of generality, a single-objective optimization problem may be expressed as:

$$\begin{aligned} &\min f(\mathbf{x}) \\ &\text{subject to } \mathbf{x} \in S \end{aligned} \tag{1}$$

with $f(\mathbf{x}) : S \rightarrow \Re$. The (nonempty) feasible region S is a subset of the decision variable space $\Re^n$ and consists of decision variable vectors $\mathbf{x} = (x_1, \ldots, x_n)^T$ that satisfy all the constraints.

Given the initial design $D = \{(\mathbf{x}^m, f(\mathbf{x}^m))\}_{m=1}^{M}$ of M samples, a Kriging model may be constructed. It is essentially a collection of random variables, and any finite number of these have a joint Gaussian distribution [37]. The predictive density of the Kriging model for an individual $\mathbf{x}$ may be expressed as:

$$P(\hat{f}(\mathbf{x})|\mathbf{x}, D, \theta) = \mathcal{N}(\mu(\mathbf{x}), \sigma(\mathbf{x})), \tag{2}$$

where the predicted mean and the variance are given by

$$\mu(\mathbf{x}) = \boldsymbol{\kappa}(\mathbf{x}, X, \theta) K^{-1} \mathbf{f} \tag{3}$$

$$\sigma(\mathbf{x}) = \kappa(\mathbf{x}, \mathbf{x}, \theta) - \boldsymbol{\kappa}(\mathbf{x}, X, \theta)^\top K^{-1} \boldsymbol{\kappa}(X, \mathbf{x}, \theta). \tag{4}$$

Here, $X \in \Re^{M \times n}$ is the design matrix (that consists of the initial LHD at the start of the algorithm and is later augmented by additional evaluations) in the decision space and $\mathbf{f} \in \Re^M$ is the vector of associated expensive function responses. The covariance matrix $K \in \Re^{M \times M}$ captures the covariances among observations as defined by (covariance or kernel) function $\kappa(\mathbf{x}', \mathbf{x}'', \theta)$ where $\mathbf{x}', \mathbf{x}'' \in X$ are two observed decision vectors. $\boldsymbol{\kappa}(\mathbf{x}, X, \theta)$ is the vector of covariances between an arbitrary decision vector $\mathbf{x}$ and the observations X. The hyperparameters θ control the nature and the flexibility of the specified kernel function. It should be noted that any function dependent on two decision vectors may be used to capture the covariances, as long as the derived matrix K remains positive semi-definite [37]. The kernel functions used in this chapter and the associated hyperparameters are described in section 4.1.2.

Irrespective of the particular kernel function used, training a Kriging model constitutes estimating the hyperparameters θ by maximizing the log likelihood of the

data given by:

$$\log P(D|\theta) = -\frac{1}{2}\log|K| - \frac{1}{2}\mathbf{f}^\top K^{-1}\mathbf{f} - \frac{M}{2}\log(2\pi). \tag{5}$$

Although it is possible to marginalize the hyperparameters using Markov Chain Monte Carlo method [42], we do not investigate its efficacy here. [2]

The predicted improvement over the current best $f^* = \min_m f(\mathbf{x}^m)$ is $I(\mathbf{x}, f^*|\hat{f}) = \max\{f^* - \hat{f}(\mathbf{x}), 0\}$. Hence, the expected improvement may be calculated as:

$$E[I(\mathbf{x}, f^*|\hat{f})] = \int_{-\infty}^{\infty} I(\mathbf{x}, f^*) P(\hat{f}|\mathbf{x}, D, \theta) d\hat{f} = \sigma(\mathbf{x})(s\Phi(s) + \phi(s)), \tag{6}$$

where, $s = \frac{f^* - \mu(\mathbf{x})}{\sigma(\mathbf{x})}$, and $\Phi(.)$ and $\phi(.)$ are the standard normal cumulative and probability density function, respectively.

Given the model, the solution that maximizes the expected improvement is expected to yield the most improvement over the best evaluated solution so far. Therefore, in the EGO framework, the next solution that is subjected to expensive evaluation is $\mathbf{x}^* = \text{argmax}_{\mathbf{x}} E[I(\mathbf{x}, f^*|\hat{f})]$. The dataset D is augmented with $\mathbf{x}^*$, i.e. $D := D \cup \{(\mathbf{x}^*, f(\mathbf{x}^*))\}$, and the Kriging model is retrained. Until the budget of expensive function evaluations is exhausted, the process is repeated. The framework is summarized in Algorithm 1.

Algorithm 1 Efficient global optimization.

Inputs

M : Number of initial samples
T : Budget on expensive function evaluations
$f(\mathbf{x})$: Expensive objective function

Steps

1: $X \leftarrow$ Latin Hypercube Sampling(S) ▷ Generate initial samples
2: $\mathbf{f} \leftarrow f(\mathbf{x} \in X)$ ▷ Expensively evaluate all initial samples
3: **for** $i = M \rightarrow T$ **do**
4: $\hat{f} \leftarrow$ Train Kriging Model$(X, \mathbf{f})$ ▷ Train a Kriging model
5: $\mathbf{x}^* \leftarrow \text{argmax}_x E[I(\mathbf{x}, f^*|\hat{f})]$ ▷ Maximize expected improvement
6: $X \leftarrow X \cup \{\mathbf{x}^*\}$ ▷ Augment data set with $\mathbf{x}^*$
7: $\mathbf{f} \leftarrow \mathbf{f} \cup \{f(\mathbf{x}^*)\}$ ▷ Expensively evaluate $\mathbf{x}^*$
8: **end for**
9: **return** $X, \mathbf{f}$

[2] It should be noted that it is common to use maximum likelihood estimation of hyperparameters rather than integrating over all possible hyperparameters given a prior probability distribution. Although some research suggest it aids the optimization process, but it may increase the overall computation time [42].

2.1 Suitability of EGO

Although the EGO framework has been successfully demonstrated in the literature, it can be envisaged that it may not be suitable for all classes of objective functions. In this chapter, we are partly interested in laying out the foundation of scrutinizing the appropriateness of EGO with respect to different problem classes through experiments. It is therefore pertinent to know what is currently considered as the advantages and disadvantages of EGO, as this helps the practitioners to determine its suitability to their specific use case. We briefly present such properties of EGO below.

Advantages

- It is a very flexible predictor, and this flexibility is borne of the specific kernel function [37].
- It is possible (and often recommended) to incorporate expert knowledge, for instance via Co-Kriging [15], or trend functions [4].
- In addition to continuous domains, kernel functions may be designed for various data representations [50].
- The Kriging posterior predictive distribution is Gaussian, and thus it permits exact computation of uncertainty based infill criteria, e.g. the expected improvement [39].
- Data that is non-deterministic, e.g. subject to measurement errors (noise), can be readily incorporated into the model. However, if the noise is heteroscedastic, then the infill criterion may no longer be appropriate [14].
- It allows automatic relevance determination, i.e. irrelevant variables may easily be discounted through the hyperparameters of the covariance function [37].
- It is possible to further improve prediction, and consequently optimization performance, by reducing the overall uncertainty in prediction using the gradient information of the expensive function (if available) [28].

Disadvantages

- The computational complexity for prediction (and training) using Kriging is $\mathcal{O}(M^3)$ for M data points due to matrix inversions (or decompositions) necessary for training and prediction [5]. Therefore, a large data set may become costly to train and to predict with. However, sparse Kriging [41], or cluster Kriging [46], where a subset of all data points are carefully selected to train models, may be used to tackle this issue.
- Kriging performs poorly for high dimensional decision spaces (e.g., $n \geq 20$). This is because a large amount of data is required to build a representative model, and also distances fail to appropriately represent proximity in higher dimensions [3]. High dimensional decision spaces may necessitate some dimensionality reduction method [43].
- Selecting a sensible kernel function requires domain expertise. Unsurprisingly, it is therefore hard to design a general framework that would work well for all kinds of problems.

- Discontinuities in the objective landscape may be problematic for Kriging. This is because standard stationary kernels (suggested for general use [42]) may fail to approximate the objective function well enough for it to be useful in EGO. Nonetheless, such kernels usually work well for smooth landscapes (given enough training data).
- Except for the relevance determination (or the sensitivity of the variables), it is difficult to derive any logical conclusions on the interactions or relationships between the decision space and the objective space, which is possible for example, from linear and decision tree models.
- Especially in the context of optimization, data-sets may become very dense in specific areas of the decision space, e.g., when candidate solutions cluster around a potential optimum. This may yield close-to singular correlation matrices, which may cause numerical problems with regards to matrix inversions or decompositions (e.g., Cholesky decomposition).

It should be noted that these advantages and disadvantages may not be valid for all variants of the Kriging method, but they capture the behavior of some of the most frequently employed implementations.

3 Integration of model and optimizer

In this section, we detail three different areas where surrogate and EA interact, and the SAEA could thus profit from more explicit integration. These are integration of: 1) search operators and kernels, 2) error measure and performance measure and 3) acquisition functions and optimizer configuration.

3.1 Integration I: Search Operators and Kernels

One important aspect of Kriging and related models is the choice of their kernels. These kernels or correlation functions are important as they essentially control how the model perceives local neighborhoods. Here, we understand neighborhoods as connected areas with similar function values. For example, a fast decaying kernel function leads to smaller neighborhoods, and vice versa. Similarly, search operators of evolutionary algorithms define the neighborhood structure that the optimization algorithm perceives. This analogy of kernels and search operators highlights why their integration may be a promising next step.

The importance of the selection of the correct search operator is well established in the literature, e.g. highlighted by studies on search operator tuning for evolutionary algorithms. For instance in [49], an empirical study has been performed to tune different genetic operators. Similarly in [13], the authors showed the influence of different search operators on the performance of the employed evolutionary algorithm. On the other hand, selection of a particular kernel or a combination of them

during optimization has also been studied in the literature. To alleviate this problem, approaches such as model ensembles [2, 27] and selecting one kernel based on the accuracy [40] exist in literature. Nevertheless, research on the integration of search operators and kernels in model based evolutionary algorithms is limited.

One promising exception is the recent study by Lane et al. [25]. They propose the use of kernels in the context of evolutionary search operators. As the very same kernels may then be employed in the optimization algorithm (here: an evolutionary algorithm) as well as the model (e.g., Kriging or SVMs) Lane et al. [25] state that this might

> *lead to a more seamless integration between EAs and kernel-based surrogate models being used to augment them.*

This further integration based on kernels seems to be quite promising. In fact, the modeling procedure might be able to suggest not just the right search operator, but as well may suggest a corresponding step size parameter value, based on the parameters of the model. This, in combination with the self-adaptive capabilities of EAs may be a profitable direction for further research.

Such potential integration of model and optimizer is not limited to EAs. Other optimization algorithms employ kernel functions, e.g. Estimation of Distribution Algorithms (EDAs) [20]. Here, a distribution is iteratively fit to estimate the location of the best candidate solutions in the decision space. Samples from the distribution are taken, and evaluated sequentially. In the context of EDAs, distributions can hence be interpreted as search operators. Clearly, the distribution may as well be based on kernel functions. However, an arbitrary new kernel would also rely on the availability of an efficient sampling technique.

3.2 Integration II: Error Measure and Performance Measures

The training process of a surrogate model integrated into an SAEA can be interpreted as an optimization problem regarding some error or other performance measure. A common choice is an error measure computed based on the predicted values or residuals, such as the mean squared error. However, the ability to distinguish between solutions is often more important than absolute prediction accuracy in the context of SAEAs. In the following, we therefore highlight two alternative concepts for performance measures that integrate better with their usage within an SAEA: rank and locality.

Many meta-heuristic optimization algorithms are based on comparisons and ranks rather than absolute objective values. Hence, they do not actually require an exact, numeric prediction provided by the surrogate model. A ranking of candidate solutions would be sufficient (cf. [45]). Machine learning models rarely consider ranking errors. There are, however, attempts at rank-based surrogate models

as described by Runarsson [38] or the approach based on rank-SVMs proposed by Loshchilov et al. [29].[3]

Besides ranking, another important issue is locality. While machine learning models are often optimized regarding global accuracy, it is clearly more important to have high accuracies in promising areas of the decision space. This holds true regardless of the nature of the prediction, i.e. function value or rank. A first step in this direction could be to bias the initial sample for the training of the surrogate model with previous knowledge (if available) instead of using a space-filling design.

In terms of local performance measures for surrogate models in SAEAs, Le et al. have recently introduced the concept of *evolvability* [26]. In this context, the *evolvability* of a surrogate model is defined as the expected amount of improvement of an offspring (derived by local search) in comparison to its parent. As such, the *evolvability* measure considers information about the fitness landscape, the state of the search process as well as aspects of the optimization algorithm [6].

Another important question about measuring of optimization performance is what kind of optimal values can really be reached under the restriction of a strictly limited budget. This also extends to the question of measuring model quality: what quality can be reached, given a potentially small, sparse data set? For instance, the convergence properties of EGO can be computed analytically [7], yet real-world restrictions will often allow so few evaluations that these theoretical considerations become pointless. Following these arguments, Wessing and Preuss [48] recently questioned whether global optimization is actually a suitable goal for EGO, or whether it is much better suited for the discovery of multiple local optima. This line of argument also affects the question of optimizer and model integration. If global optimization is not the goal, the model does not need to be accurate enough to respect the global structure, and the optimization algorithm will not necessarily have to find the exact optimum of the surrogate. Hence, with respect to the problem definition, the required amount of accuracy of both the model and the optimizer should be traded off against their cost in a more controlled manner.

3.3 Integration III: Acquisition Functions and Optimizer

EGO is primarily driven by a utility function that indicates the usefulness of a candidate solution that may be subjected to expensive evaluation. This utility function is often referred to as the infill criterion, updating criterion or acquisition function. This, in essence, forms the connection between the model and the optimizer. Thus, one critical issue is how to determine a good acquisition function that is capable of utilizing the information from the model and provide a good balance between exploration and exploitation.

[3] Rank models are also important in the domain of multi-objective optimization. Here, ranks can be easily produced (via non-dominated sorting), whereas numeric indicators (such as crowding distance or hypervolume) are still subject of current research. Rank-based models would allow to represent multiple objectives with just one single surrogate model.

Many different acquisition functions have been proposed in the literature, e.g., probability of improvement (PI), expected improvement (EI), lower confidence bound (LCB), stepwise uncertainty reduction (UR), etc.; for more details see [39]. While some, like EI, have proven convergence properties [7], the characteristics, e.g., multi-modality of an acquisition function are vital in the performance of the overall algorithm.

To enable further integration between modeling and optimization, we may exploit the knowledge available regarding the acquisition function. Unlike the real (black-box) objective function, some of the properties of the infill criterion are usually known: they depend on the model as well as the criterion itself. Such knowledge could be more explicitly exploited by the optimization algorithm. One example of such properties is the multi-modal landscape of the function. By design, most uncertainty based acquisition functions promote search away from the observed solutions. As such, they naturally generate a multi-modal landscape with local optima that reside between observed solutions, as shown in Fig. 2. This information may be used

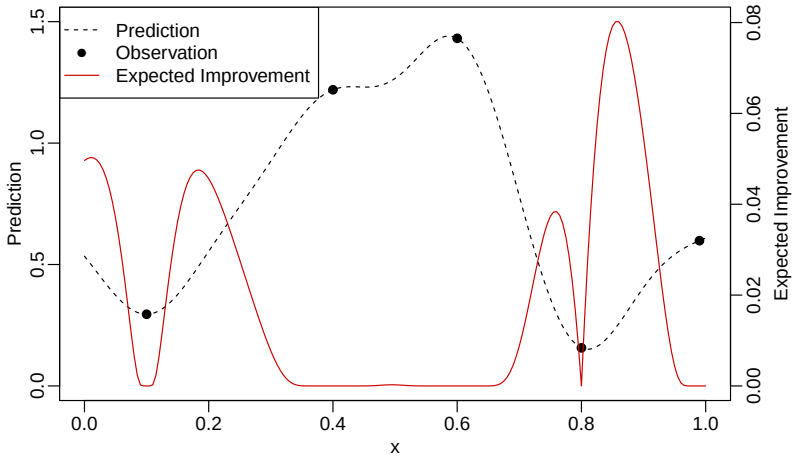

Fig. 2 An example of the prediction and expected improvement derived from a Kriging model. It showcases the potential multi-modality of the expected improvement fitness landscape. This examples assumes minimization of the objective function (and hence, the prediction) but maximization of the expected improvement.

to tune parameters of the optimizer that are strongly related to the multi-modality of a problem, e.g., the number of restarts, or the population size of an EA. Also, initializing optimizers by generating new samples between known solutions may be helpful. A similar integration step can be made on the side of the optimizer: here, the properties are known and can be exploited.

As previously mentioned in Sect. 3.2, most optimizers do not require exact objective function values, but rather, are based on comparing candidate solutions. The development of rank-based infill criteria may hence be a promising research direction. When a model predicts ranks rather than numeric predictions, criteria such

as expected improvement would automatically be reinterpreted as rank-based measures.

Another possible avenue of research is to interpret the two conflicting components that are balanced in most acquisition functions, i.e. exploration and exploitation, as two separate goals [44, 47]. In this case, evolutionary algorithms specifically designed for multi-objective optimization are employed to potentially find more balanced solutions.

4 Numerical experiments

In this section, we first briefly elaborate different types of kernels, recombination and mutation operator, and their combinations used. In addition, we give an introduction to benchmark problems used in this study. Then, we present the results and discuss them based on several experiments conducted.

4.1 Experimental Setup

4.1.1 Algorithm Configurations

As stated in the research questions in Sect. 1, we are interested in the effects and interactions of i) surrogate model, ii) optimization algorithm and iii) optimization problem. For the surrogate model, we investigate the influence of different kernels. For the optimization algorithm, we consider different variation operators in an evolutionary algorithm (EA) which are combined in an EGO variant. The parameter values of the different modules of our algorithms are given in Table 1. As a baseline comparison, we also conducted the experiments with a model-free EA (genetic algorithm in this case) with a population of size $2 \times n$, with Gaussian mutation operator and uniform crossover. For the model-based EGO, we also tested different initial design sizes when training the Kriging model. Note, that when the initial design size is equal to the whole budget of evaluations, EGO reduces to Latin Hypercube sampling (LHC).

We use the python GPy [1] library for the Kriging implementation. GPy encodes Gaussian process regression and offers several different kernels to be used in the model. For the optimizer, we use the DEAP library [16], which is a modular framework for evolutionary algorithms. For details on the employed modeling and optimization tools, we refer to the documentation.

The acronyms mentioned in the table are defined as follows:

- n_samples: initial design size, the number of samples used to build the first Kriging model in each run
- budget: maximum number of objective function evaluations allowed to the algorithm, including the initial design sampling

Table 1 Parameter values of different modules considered in this study. The first column specifies the component, the second column the parameter name and the third column gives the chosen parameter values, as well as additional, related values and further details. The last column specifies the data type of the parameters, i.e., whether they are integer (int.), or categorical (cat.).

Component	Name	Details	Type
Main	*n_samples*	0.25 × budget 0.5 × budget 1.0 × budget	int.
	budget	$n \times 20$	int.
	method	EA	cat.
Model	Kernels	see 4.1.2	cat.
EA	selection	tournament (size: 3)	cat.
	mutation	multi-variate Gaussian, σ = diag(0.1), mutpb = 0.1 polynomial bounded, eta=20, indpb = 1/n mutESLogNormal, c=20, indpb = 1/n	cat.
	recombination	uniform, indpb = 0.5, cxpb = 0.8 simulated binary bounded, eta =20,cxpb = 0.8	cat.
	budget_surrogate	4000 × n	int.
	pop_size	0.01 × budget_surrogate	int.

- method: optimizer or the evolutionary algorithm used
- kernels: different correlation functions in the Kriging model. Details of the chosen kernels are introduced in 4.1.2.
- selection: selection operator which is kept fixed to tournament selection operator with size 3 in this study
- mutation: mutation operator
- recombination: crossover or recombination operator
- budget_surrogate: number of function evaluations used to optimize the surrogate model with the optimizer (in each iteration)
- pop_size: population size in the EA
- mutpb: probability of mutation
- cxpb: probability of recombination
- indpb: parameter of mutation and recombination operators
- eta: distribution index of mutation and recombination operators
- c: learning parameter in mutESLogNormal mutation operator
- σ: variance, step size of the mutation operator

The chosen kernels are excluded from Table 1. Instead, since they require additional details and explanations, they are introduced in the next section.

4.1.2 Kernels

In essence, a kernel function encapsulates the relationship and the permitted variation in function responses between two decision vectors $\mathbf{x}$ and $\mathbf{x}'$. A kernel with its hyperparameters thus imposes a reproducing kernel Hilbert space for all possible functions that may be represented. A typical avenue to describe the relationship is through a distance measure $r^2 = \sum_{i=1}^{n}(x_i - x_i')^2/l_i^2$, where l_i is a hyperparameter which determines the lengthscale associated with the ith dimension. Here the role of l_i is to scale, and consequently specify the importance of the ith dimension in the decision space with respect to the function responses. In addition, another hyperparameter often used is the kernel variance σ_k that controls the amplitude of respective kernels and determines how much the function responses may vary depending on distances. Hence, the set of hyperparameters $\theta = \{\sigma_k, l_1, \ldots, l_n\}$ effectively control what function responses may be achieved with Kriging models. As such, learning in this context constitutes locating a set of suitable hyperparameters that represents the data best[4]. In this chapter, we used the following kernel functions [1, 37]:

- **RBF:** The Gaussian or radial basis function (RBF) kernel is the most popular kernel. It has infinitely many derivatives and it is a universal (stationary) kernel.

$$\kappa(\mathbf{x}, \mathbf{x}') = \sigma_k^2 \exp\left(-\frac{r^2}{2}\right), \tag{7}$$

- **MLP:** The multi-layer perceptron (MLP) is a class of kernel functions that captures the flexibility of multi-layer neural networks with infinitely many hidden units. It is a non-stationary kernel. It is also known as the arc-sine or neural network kernel.

$$\kappa(\mathbf{x}, \mathbf{x}') = \sigma_k^2 \frac{2}{\pi} \operatorname{asin}\left(\frac{\boldsymbol{\sigma}_w^2 \mathbf{x}^T \mathbf{x}' + \sigma_b^2}{\sqrt{\boldsymbol{\sigma}_w^2 \mathbf{x}^T \mathbf{x} + \sigma_b^2 + 1}\sqrt{\boldsymbol{\sigma}_w^2 \mathbf{x}'^T \mathbf{x}' + \sigma_b^2 + 1}}\right), \tag{8}$$

  where $\boldsymbol{\sigma}_w$ and σ_b hyperparameters can be seen as the variances in weight vector and a bias respectively for a neural network with infinitely many hidden layers.
- **EXP:** The exponential (EXP) kernel is another stationary kernel, closely related to the RBF kernel.

$$\kappa(\mathbf{x}, \mathbf{x}') = \sigma_k^2 \exp(-r), \tag{9}$$

- **Mat52:** The Matern-5/2 is a stationary kernel that is twice differentiable. It is frequently recommended to use this kernel for real world problems [42].

$$\kappa(\mathbf{x}, \mathbf{x}') = \sigma_k^2 \left(1 + \sqrt{5}r + \frac{5}{3}r^2\right) \exp\left(-\sqrt{5}r\right), \tag{10}$$

- **LIN:** The linear kernel is a non-stationary kernel that captures any linearity presented by data.

[4] We used limited memory BFGS with five restarts to estimate the hyperparameters [1].

$$\kappa(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{n} \sigma_i x_i x_i', \tag{11}$$

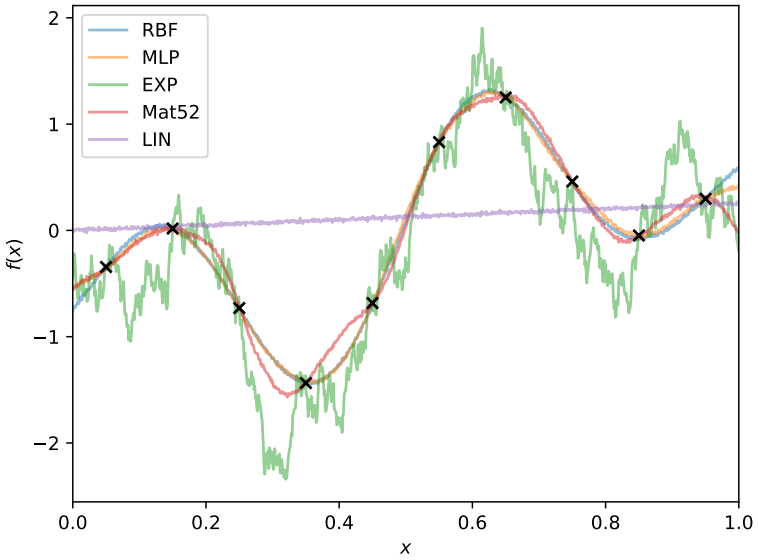

Fig. 3 Randomly drawn realizations of Gaussian processes with different kernels. The hyperparameters are optimized based on a hypothetical set of data shown in black crosses. Clearly, EXP produces a non-smooth realization and LIN only captures the linear trend in the data. In contrast, RBF, MLP, and Mat52 generate smoother representations of the data.

To understand these kernels, it is helpful to investigate how realizations of the respective models behave. It is common to depict the mean prediction and the uncertainty around it due to specified decision vectors. Generally, such visualizations smooth over how a realization from the full posterior predictive distribution may behave. Thus, it is somewhat difficult to observe differences in behavior. Therefore, it is useful to generate a random multi-variate normal sample for a range of decision vectors at regular intervals. In Fig. 3, we show such random realizations of the above kernels. Unsurprisingly, the realizations clearly suggest that it is unlikely to derive good performance from a specific kernel across problems with different characteristics. For instance, a linear kernel may only be useful if the expensive function is in fact linear. Or, a fast changing and non-smooth function may be better represented with an exponential kernel.

4.1.3 BBOB benchmark problems

The BBOB function suite within the benchmarking COCO framework contains 24 different single-objective functions, which are scalable regarding the decision space

dimension. These functions vary in terms of characteristics such as separability, conditioning, modality and global structure, thus ensuring some level of diversity.

In order to be able to formulate statements on the performance of an algorithm on a function type as well as its robustness, the functions can be instantiated. The test suite contains 15 instances for each of the functions that result from transformations and differ in terms of some properties, e.g., the location of optima. In our experiments, we considered problems with decision space dimensions 2, 3, 5 and 10. The bounds of the variables are also limited to $[-5,5] \subset \Re$ per dimension. We set the maximum budget of function evaluations to only 20 evaluations per dimension.

Since the optima are known for each function instance, the performance of an algorithm can be measured as the difference of the best discovered value and the correct optimum (precision). The benchmark measures anytime performance by recording the best precision achieved at each function evaluation. To that end, precision targets are defined and it is recorded when the algorithm is able to reach the respective target.

4.1.4 Summary of Test Runs

With the above described choices the following three (partially overlapping) sets of experiments were conducted:

1. First, the kernel (RBF), mutation operator (Gaussian) and recombination operator (Uniform) were kept fixed, while the initial design size was set to either 25% ($5n$) or 50% ($10n$) of the budget ($20n$), where n is the dimension of the decision space.
2. Second, the interaction of kernel choice and mutation operator was investigated, by testing all combinations of kernels and mutation operators. The initial design size was fixed at 25% ($5n$).
3. Finally, only for the RBF and MLP kernel, all combinations of mutation and recombination operators were tested. The initial design size was fixed at 25% ($5n$).

All tests were run on the complete BBOB test suite with specification as detailed in the previous section.

4.2 Results

To analyze the results of the experiments, a measure of performance is required. Following the BBOB framework, we compute empirical cumulative distribution functions (ECDF) [18]. Hence, we specify a set of target values (in terms of precision) that should be achieved by the tested algorithms. Since we do runs with severely limited budgets, we do not use the default targets of BBOB, but only a set of easier targets: tar $= \{10^3, 10^{2.8}, 10^{2.6}, ..., 10^{-3}\}$. The runtimes required to attain these targets are recorded for each algorithm run and target. As an aggregated measure

of performance, the Area Under the resulting ECDF Curves (AUC) is computed for each algorithm run.

As outlined earlier, we are interested in interactions of the problem (i.e., represented by type of function or dimensionality), the optimizer (mutation and recombination operators) and the model (kernel). To discover whether any of these interactions are observed in the experimental results, we use an ANOVA / linear regression analysis [32]. All main effects and two-way interaction terms are included in the model.

First experiments are focused on one kernel (RBF), mutation operator (Gaussian) and recombination operator (Uniform), but with varying initial design sizes (25 and 50 percent of the budget). The resulting linear model has an R^2 value of approximately 0.9262. This indicates that a large proportion of the variance in the observations is explained by the model. It has to be noted, that the ANOVA model assumes independent random samples from a normal distribution with constant variance (homoscedastic).

As the analysis plots in Fig. 4 indicate, the residuals are not perfectly normally distributed (due to their deviation from the diagonal in the quantile-quantile plot (QQ-plot), right side in Fig. 4) and may be slightly heteroscedastic (due to the structure seen in the residuals, left side in Fig. 4). Still, as the deviations are not that extreme and since the number of observed samples is rather large, we argue that the model is still adequate. Similar results are valid for all models described in this section.

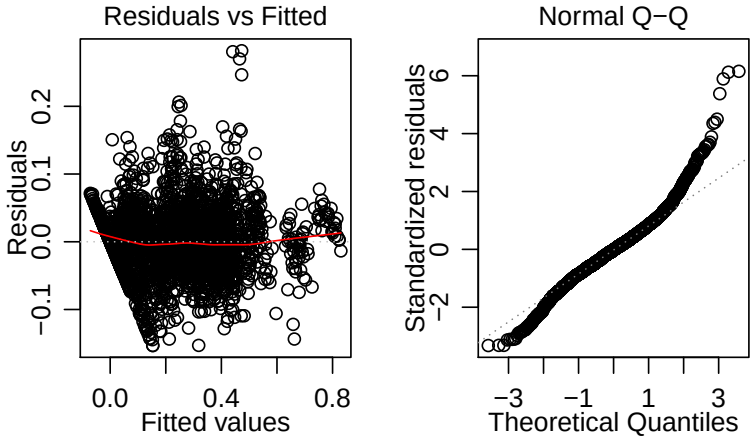

Fig. 4 Analysis plots for checking the assumptions (normal distribution, homoscedastic) of the model. This is the result for the data generated with fixed kernel, fixed mutation and recombination operators, and varying design sizes. All main effects and interactions are considered in the model. The plot is based on the residuals of the model, plotting them against the fitted values (left) and plotting their actual quantiles against the theoretical (normal distribution) quantiles

The resulting ANOVA for the first set of data is presented in Table 2. The ANOVA

Table 2 ANOVA for preliminary tests with RBF kernel, Gaussian mutation and Uniform recombination operator, with varying initial design sizes. The rows report statistics for target function (fun), dimension (dim), function instance (inst), design size (size) and their respective interaction terms. Importantly, numbers close to zero (bold numbers, < 0.05) indicate significant effects. For an interpretation of other reported statistics, see [32].

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
fun	23	58.64	2.55	1030.47	**<2.2e-16**
dim	1	11.75	11.75	4748.94	**<2.2e-16**
inst	14	0.04	0.00	1.05	0.3952
size	1	0.20	0.20	80.80	**<2.2e-16**
fun:dim	23	4.11	0.18	72.25	**<2.2e-16**
fun:inst	322	0.59	0.00	0.74	0.9998
fun:size	23	0.45	0.02	7.88	**<2.2e-16**
dim:inst	14	0.02	0.00	0.63	0.8412
dim:size	1	0.07	0.07	28.87	**8.467e-08**
inst:size	14	0.02	0.00	0.46	0.9521
Residuals	2443	6.04	0.00		

determines that function, dimension, initial design size and their interactions seem to have significant effects. The problem instance and its corresponding interaction terms have no significant effect. This is a promising first result: Different functions and dimensions should affect performance, but an instance of the same problem should not. The effect of the initial design size is also easy to explain: a smaller size leaves a larger part of the budget for a more purposeful exploration of the decision space. While final best values (after the budget is exhausted) may not vary much, runs with smaller design sizes seem to reach good solutions faster.

AUC values are overall larger for all runs with the smaller of the tested initial design sizes (25 percent of the budget). Note, that this general remark is not affected by the interaction terms. The respective coefficients (which we do not include due to their large number) indicate that the design size has stronger effects for certain functions. The rest of the analysis will hence focus on results with the lower design size and the insignificant influence of test function instances will be ignored.

The second set of experiments tested all 5 different kernels in the model in combination with 3 different mutation operators employed in the optimizer. The resulting linear model has an R^2 value of approximately 0.9034. Again, this indicates that a large proportion of the variance in the observations is explained by the model. The ANOVA for this model is presented in Table 3. While all other terms are significant, the interaction of kernel and mutation operator is insignificant. There seems to be no evidence that changing both kernel and mutation operator has an effect on the optimization performance. Changing them individually however does have a significant effect, there are even multiple significant interactions with the test function and dimension. That means, the performance of different kernels changes depending on the objective function.

The third set of experiments was intended to determine interactions with the recombination operator. Hence, all combinations of two kernel choices (RBF, MLP) three mutation operators, and two recombination operators were tested. The result-

Table 3 ANOVA for tests with different kernels and mutation operators. The rows report statistics for target function (fun), dimension (dim), function instance (inst), kernel (kern), mutation operator (mutation) and interaction terms. Importantly, numbers close to zero (bold numbers, < 0.05) indicate significant effects. For an interpretation of the other reported statistics, see e.g., [32].

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
kern	4	0.99	0.25	90.00	**<2.2e-16**
mutation	2	0.23	0.11	41.22	**<2.2e-16**
fun	23	430.33	18.71	6792.26	**<2.2e-16**
dim	1	85.30	85.30	30965.08	**<2.2e-16**
kern:mutation	8	0.02	0.00	0.87	0.5413
kern:fun	92	1.37	0.01	5.40	**<2.2e-16**
kern:dim	4	1.00	0.25	91.03	**<2.2e-16**
mutation:fun	46	1.93	0.04	15.24	**<2.2e-16**
mutation:dim	2	0.03	0.01	4.67	**0.0094**
fun:dim	23	29.76	1.29	469.79	**<2.2e-16**
Residuals	21394	58.93	0.00		

ing linear model has an R^2 value of approximately 0.9131. This indicates that a large proportion of the variance in the observations is explained by the model. The ANOVA for this model is presented in Table 4. Again, the interactions between the

Table 4 ANOVA table, analyzing influence of the used kernel (kern), mutation operator (mutation), recombination operator (operator), test function (fun), dimension (dim) and their interactions. Importantly, numbers close to zero (bold numbers, < 0.05) indicate significant effects. For an interpretation of other reported statistics, see [32].

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
kern	1	0.01	0.01	4.29	**0.0384**
mutation	2	0.09	0.04	16.22	**9.126e-08**
recombination	1	0.20	0.20	74.52	**<2.2e-16**
fun	23	365.71	15.90	6056.62	**<2.2e-16**
dim	1	77.02	77.02	29339.03	**<2.2e-16**
kern:mutation	2	0.00	0.00	0.45	0.6407
kern:recombination	1	0.00	0.00	1.50	0.2214
kern:fun	23	0.32	0.01	5.30	**2.292e-15**
kern:dim	1	0.01	0.01	3.99	**0.0457**
mutation:recombination	2	0.01	0.01	2.60	0.0744
mutation:fun	46	1.12	0.02	9.30	**<2.2e-16**
mutation:dim	2	0.01	0.00	1.43	0.2398
recombination:fun	23	0.63	0.03	10.38	**<2.2e-16**
recombination:dim	1	0.00	0.00	1.35	0.2457
fun:dim	23	27.60	1.20	457.17	**<2.2e-16**
Residuals	17127	44.96	0.00		

variation operators (mutation, recombination) and the kernels are not significant.

The analysis presented above determined which parameters of the algorithm configurations affect performance. But it remains unclear, which algorithm configuration works well, and on what type of function. To that end, as a follow up on the ANOVA analysis, we use Tukey's Honest Significant Difference (Tukey's

HSD) test [32, Sec. 3-5.7] to generate pairwise comparisons of all algorithm configurations, with respect to their performance on different test functions, function instances, and dimensions. The resulting pairwise comparisons are used to compute non-dominated sorting ranks: all algorithms that were not significantly worse than any other algorithm received rank one, and were removed from the list. Then, all of the remaining algorithms that were not significantly worse than any other algorithm received rank two. This procedure is iterated until all algorithms are ranked. The ranking is done for all functions, as well as each function class (separable, low/moderate conditioning, high conditioning, multi-modal with strong global structure, weak global structure). See [19] for details on the function classes and the respective properties. Table 5 presents the respective ranks. Variants with RBF or MLP kernel and Simulated Binary recombination seemed to perform best overall. Both model-free base-line algorithms (EA and LHC) performed worst. Of all model-based variants, the ones with linear kernel performed worst. In case of the multi-modal functions with weak global structure, the chosen algorithm configuration did not matter much, there were no differences between the model-based approaches. This may be attributed to the larger difficulty of these functions, due to the weak global structure.

Table 5 Ranks and AUC values for each algorithm on all functions (All), separable functions (A), low to medium conditioned functions (B), high conditioning functions (C), multi-modal functions with adequate global structure (D), and multi-modal functions with weak global structure (E). Left part shows ranks based on Tukey's HSD tests, right part shows average AUC values for the respective groups. Sorted by the AUC column *All*.

	All	A	B	C	D	E	All	A	B	C	D	E
RBF.Gauss.SimulatedBinary	1	1	1	1	1	1	0.264	0.330	0.201	0.158	0.313	0.306
Mat52.Gauss.Uniform	1	1	1	1	1	1	0.263	0.321	0.206	0.161	0.311	0.306
RBF.LogNorm.SimulatedBinary	1	1	1	1	1	1	0.261	0.311	0.204	0.157	0.314	0.307
MLP.Gauss.SimulatedBinary	1	1	1	2	1	1	0.261	0.324	0.203	0.144	0.313	0.308
RBF.Poly.SimulatedBinary	1	1	1	1	1	1	0.261	0.311	0.208	0.153	0.315	0.306
MLP.LogNorm.SimulatedBinary	1	2	1	1	1	1	0.259	0.304	0.211	0.147	0.315	0.311
RBF.Gauss.Uniform	1	1	1	1	1	1	0.259	0.318	0.204	0.152	0.311	0.300
MLP.Poly.SimulatedBinary	1	2	1	2	1	1	0.258	0.308	0.206	0.144	0.317	0.303
MLP.Gauss.Uniform	1	1	1	1	1	1	0.257	0.312	0.201	0.145	0.314	0.303
EXP.Gauss.Uniform	1	2	1	1	1	1	0.257	0.306	0.199	0.147	0.317	0.304
Mat52.Poly.Uniform	2	3	1	1	1	1	0.254	0.288	0.198	0.152	0.318	0.303
RBF.LogNorm.Uniform	2	2	1	1	1	1	0.253	0.289	0.205	0.148	0.309	0.303
Mat52.LogNorm.Uniform	2	3	1	1	1	1	0.253	0.285	0.198	0.149	0.316	0.305
MLP.LogNorm.Uniform	2	3	1	2	1	1	0.252	0.284	0.202	0.141	0.314	0.309
EXP.LogNorm.Uniform	2	3	1	2	1	1	0.252	0.278	0.203	0.141	0.319	0.308
MLP.Poly.Uniform	2	3	1	2	1	1	0.251	0.285	0.205	0.140	0.313	0.305
RBF.Poly.Uniform	2	3	1	1	1	1	0.251	0.282	0.201	0.150	0.309	0.301
EXP.Poly.Uniform	2	3	1	2	1	1	0.250	0.277	0.202	0.141	0.317	0.302
LIN.Gauss.Uniform	3	3	2	2	2	1	0.241	0.288	0.176	0.128	0.302	0.298
LIN.LogNorm.Uniform	3	4	2	2	2	1	0.237	0.261	0.182	0.131	0.302	0.296
LIN.Poly.Uniform	3	4	2	2	2	1	0.235	0.257	0.177	0.130	0.307	0.292
LHC	4	5	3	3	2	2	0.214	0.205	0.152	0.111	0.303	0.284
EA.Gauss.Uniform	5	5	4	3	3	2	0.200	0.188	0.129	0.102	0.294	0.275

4.3 Discussion of Results

To summarize, we can draw the following conclusions from the presented results.

Should we use SAEA algorithms such as EGO? Based on the results this can be answered positively. Table 5 shows that even the worst model-based algorithm variants outperform the model-free competitors. Since this conclusion is based on the specific budget used here, they may not be generalized for larger budget.

What kernels should be used? The choice of kernel has a significant effect, yet it is hard to make a general recommendation since the best working kernel may be problem dependent. Also, several kernels may provide equally good results. The only general (yet also obvious from Fig. 3) recommendation is to avoid using the linear kernel. The earlier ANOVA indicates that the kernel choice has interactions with the objective function and its dimensionality. This means that tuning is required to find the right kernel for a specific problem.

Which search operators should be chosen? In some cases, simulated binary crossover seems to perform better than uniform crossover. While the algorithm is sensitive to the mutation operator, no general recommendation can be made. This, and the observed interactions with objective function and dimensionality clearly necessitates to be resolved by tuning.

How large should the initial design be? The smaller initial design size (25% of the budget or $5n$ for n-dimensional decision space enables the algorithm to spend more evaluations on exploiting the information provided by the model. This seems to yield better performance.

Is there an interaction between search operators and kernels? Unfortunately, no significant relationship between search operators and kernels was observed. Firstly, this could be an effect of over-searching. If we assume that the budget of surrogate model evaluations available to the optimizer is large, then the exact configuration of the optimizer will matter less, since even a poorly configured optimizer may provide sufficient results. Otherwise, the reason may be found in the selection of kernels and search operators. If none of them interact well, results are all equally poor. If all of them interact well, results will correspondingly be equally good.

Does the problem class play a role? Table 5 shows that the problem class clearly matters: depending on the amount of local or global structure, different algorithms (or sets of algorithms) may perform best. Also, if global problem structure is weak, there seems to be no difference in performance between the employed modeling procedures. Then, even the linear kernel provides competitive results.

5 Conclusions and Outlooks

In this chapter, we discussed a better integration of model and optimizer in SAEAs. We have performed an extensive study to investigate the effects and interactions of different modules of SAEAs. Most importantly, we focused on different types of kernels, search operators and their combinations. These elements of model and optimizer were applied in the EGO algorithm. The EGO variants arising from this setup were tested in several numerical experiments based on a set of single-objective test functions, taken from the BBOB framework.

The observed results provide some important insights which may be of use when developing or applying a SAEA to solve an expensive optimization problem; the effect of choosing different kernels may be affected by the dimensionality of the problem. In practice, problem dimensionality should hence be taken into consideration when choosing a kernel. Similarly, the type of objective function has a strong impact. While search operators and kernels each affect the performance of the algorithm, no interaction could be observed. The effect of search operators and kernels also depends on the objective function. Finally, smaller initial design sizes are preferable to allow sufficient evaluations to the model-driven search procedure.

In addition to these findings, statistical test procedures were used to rank the employed algorithm configurations. The ranking provides an insight to the practitioners and optimization algorithm analyst to select an appropriate element before developing or applying a SAEA; as far as problems similar to the BBOB test benchmarks are concerned, it seems to be recommendable to use RBF, Matern or MLP kernels, which (overall) seemed to work best in combination with simulated binary crossover.

The results of the experiments seem to raise the issue that a close interaction of optimizer and surrogate model is not always observable. One likely reason is the selection of kernels and operators. One way to alleviate this would be to design additional experiments where search operators and kernels are selected (by prior knowledge), so that certain combinations complement each other whereas others are opposing. That means, two extreme cases may then be of interest: i) the ideal case, combining kernels with specific, matching variation operators and ii) the worst case, search operators that are based on a completely different structure as used by the kernel. Thus, we propose that the analysis of more narrow, yet focused test cases would be an interesting next step.

This study stresses the importance looking into the integration of surrogate and optimizer. In an ideal scenario, all the components of an SAEA are well integrated and self-adaptive. For instance, selection of search operators and their corresponding parameters, selection of a kernel, using an error measure and acquisition function can be done on the fly when doing optimization. However, making rules for such a self-adaptive framework may not be straightforward and the whole procedure can be potentially time-consuming. Such self-adaptive configuration may introduce another set of parameters which could be sensitive for the performance of the algorithm.

In the future, we would like to extend our study to multi-objective optimization problems by employing state-of-the-art algorithms such as ParEGO [24], SMS-EGO [35], K-RVEA [8], and HypI [36]. In addition, we will also investigate the integration of other surrogate techniques with further optimizers like CMA-ES and differential evolution. Furthermore, there exist separate integration methods for multiple machine learning models (ensemble techniques) and optimization algorithms or their configurations, respectively. Applying these approaches and results to the integration of surrogate model and optimization algorithm is hence an interesting next step.

Acknowledgements

Parts of this work are the result of the discussions at the Surrogate-Assisted Multi-Criteria Optimization (SAMCO) Lorentz Center Workshop in Leiden, NL (February 29, 2016 till March 4, 2016). The research of Tinkle Chugh was supported by the FiDiPro project DeCoMo funded by Tekes: Finnish Funding Agency for Innovation and Natural Environment Research Council [grant number NE/P017436/1]. Alma Rahat was supported by the Engineering and Physical Sciences Research Council, UK [grant number EP/M017915/1]. The research of Martin Zaefferer is part of a project that has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 692286.

References

1. GPy: A gaussian process framework in python. http://github.com/SheffieldML/GPy (since 2012)
2. Acar, E., Rais-Rohani, M.: Ensemble of metamodels with optimized weight factors. Structural and Multidisciplinary Optimization **37**(3), 279–294 (2009)
3. Aggarwal, C.C., Hinneburg, A., Keim, D.A.: On the surprising behavior of distance metrics in high dimensional space. In: Database Theory — ICDT 2001, pp. 420–434. Springer Science + Business Media (2001)
4. Andrianakis, I., Vernon, I.R., McCreesh, N., McKinley, T.J., Oakley, J.E., Nsubuga, R.N., Goldstein, M., White, R.G.: Bayesian history matching of complex infectious disease models using emulation: A tutorial and a case study on hiv in uganda. PLoS Computational Biology **11**(1), 1–18 (2015)
5. Barber, D.: Bayesian reasoning and machine learning. Cambridge University Press (2012)
6. Bartz-Beielstein, T.: A survey of model-based methods for global optimization. In: G. Papa, M. Mernik (eds.) Bioinspired Optimization Methods and their Applications, pp. 1–18 (2016)
7. Bull, A.D.: Convergence rates of efficient global optimization algorithms. Journal of Machine Learning Research **12**, 2879–2904 (2011)
8. Chugh, T., Jin, Y., Miettinen, K., Hakanen, J., Sindhya, K.: A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization. IEEE Transactions on Evolutionary Computation **22**, 129 – 142 (2018)

9. Chugh, T., Sindhya, K., Hakanen, J., Miettinen, K.: A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. Soft Computing **23**(9), 3137–3166 (2019)
10. Coello, C., Lamont, G., Veldhuizen, D.: Evolutionary Algorithms for Solving Multi-objective Problems, 2nd edn. Springer, New York (2007)
11. Couckuyt, I., Deschrijver, D., Dhaene, T.: Fast calculation of multiobjective probability of improvement and expected improvement criteria for Pareto optimization. Journal of Global Optimization **60**, 575–594 (2014)
12. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Wiley, Chichester (2001)
13. Eiben, A., Smit, S.: Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm and Evolutionary Computation **1**(1), 19 – 31 (2011)
14. Forrester, A., Sobester, A., Keane, A.: Engineering design via surrogate modelling. John Wiley & Sons (2008)
15. Forrester, A.I., Sóbester, A., Keane, A.J.: Multi-fidelity optimization via surrogate modelling. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **463**(2088), 3251–3269 (2007)
16. Fortin, F.A., De Rainville, F.M., Gardner, M.A., Parizeau, M., Gagné, C.: DEAP: Evolutionary algorithms made easy. Journal of Machine Learning Research **13**, 2171–2175 (2012)
17. Hansen, N.: Compilation of results on the 2005 CEC benchmark functions (2005)
18. Hansen, N., Auger, A., Mersmann, O., Tušar, T., Brockhoff, D.: COCO: A platform for comparing continuous optimizers in a black-box setting. ArXiv e-prints, arXiv:1603.08785 (2016)
19. Hansen, N., Finck, S., Ros, R., Auger, A.: Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. Research Report RR-6829, INRIA (2009)
20. Hauschild, M., Pelikan, M.: An introduction and survey of estimation of distribution algorithms. Swarm and Evolutionary Computation **1**(3), 111–128 (2011)
21. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration (extended version). Tech. Rep. TR-2010-10, University of British Columbia, Department of Computer Science (2010)
22. Jin, Y.: Surrogate-assisted evolutionary computation: Recent advances and future challenges. Swarm and Evolutionary Computation **1**(2), 61–70 (2011)
23. Jones, D., Schonlau, M., Welch, W.: Efficient global optimization of expensive black-box functions. Journal of Global Optimization **13**, 455–492 (1998)
24. Knowles, J.: ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Transactions on Evolutionary Computation **10**, 50–66 (2006)
25. Lane, F., Azad, R.M.A., Ryan, C.: Principled evolutionary algorithm search operator design and the kernel trick. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1–9 (2016). DOI 10.1109/SSCI.2016.7850204
26. Le, M.N., Ong, Y.S., Menzel, S., Jin, Y., Sendhoff, B.: Evolution by adapting surrogates. Evolutionary Computation **21**(2), 313–340 (2013)
27. Lim, D., Jin, Y.: Generalizing surrogate-assisted evolutionary computation. IEEE Transactions on Evolutionary Computation **14**, 329–354 (2010)
28. Lockwood, B.A., Anitescu, M.: Gradient-enhanced universal kriging for uncertainty propagation. Nuclear Science and Engineering **170**(2), 168–195 (2012)
29. Loshchilov, I., Schoenauer, M., Sebag, M.: Dominance-Based Pareto-Surrogate for Multi-Objective Optimization. In: R.T. et al. (ed.) Simulated Evolution and Learning (SEAL 2010), pp. 230–239. LNCS 6457, Springer Verlag (2010)
30. Mckay, M., Beckman, R., Conover, W.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **42**, 55–61 (2000)
31. Mogilicharla, A., Chugh, T., Majumder, S., Mitra, K.: Multi-objective optimization of bulk vinyl acetate polymerization with branching. Materials and Manufacturing Processes **29**, 210–217 (2014)

32. Montgomery, D.C.: Design and Analysis of Experiments, 5th Edition. Wiley (1997)
33. Moscato, P., Cotta, C.: A Gentle Introduction to Memetic Algorithms, pp. 105–144. Springer US, Boston, MA (2003)
34. Oyama, A., Okabe, Y., Shimoyama, K., Fujii, K.: Aerodynamic multiobjective design exploration of a flapping airfoil using a navier-stokes solver. Journal of Aerospace Computing, Information, and Communication **6**, 256–270 (2009)
35. Ponweiser, W., Wagner, T., Biermann, D., Vincze, M.: Multiobjective optimization on a limited budget of evaluations using model-assisted S-metric selection. In: Proceedings of the Parallel Problem Solving from Nature-PPSN X, pp. 784–794. Springer, Berlin, Heidelberg (2008)
36. Rahat, A.A.M., Everson, R.M., Fieldsend, J.E.: Alternative infill strategies for expensive multi-objective optimisation. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 873–880. ACM (2017)
37. Rasmussen, C.E., Williams, C.K.I.: Gaussian processes for machine learning. The MIT Press (2006)
38. Runarsson, T.P.: Ordinal regression in evolutionary computation. In: T.P. Runarsson, H.G. Beyer, E. Burke, J.J. Merelo-Guervós, L.D. Whitley, X. Yao (eds.) Parallel Problem Solving from Nature - PPSN IX: 9th International Conference, Reykjavik, Iceland, September 9-13, 2006, Proceedings, pp. 1048–1057. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
39. Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., de Freitas, N.: Taking the human out of the loop: A review of Bayesian optimization. Proceedings of the IEEE **104**(1), 148–175 (2016)
40. Singh, H., Ray, T., Smith, W.: Surrogate assisted simulated annealing (SASA) for constrained multi-objective optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1–8. IEEE (2010)
41. Snelson, E., Ghahramani, Z.: Sparse gaussian processes using pseudo-inputs. Advances in neural information processing systems **18**, 1257 (2006)
42. Snoek, J., Larochelle, H., Adams, R.P.: Practical Bayesian optimization of machine learning algorithms. In: Advances in neural information processing systems, pp. 2951–2959 (2012)
43. Tripathy, R., Bilionis, I., Gonzalez, M.: Gaussian processes with built-in dimensionality reduction: Applications to high-dimensional uncertainty propagation. Journal of Computational Physics **321**, 191 – 223 (2016). DOI https://doi.org/10.1016/j.jcp.2016.05.039
44. Ursem, R.K.: From Expected Improvement to Investment Portfolio Improvement: Spreading the Risk in Kriging-Based Optimization, pp. 362–372. Springer International Publishing, Cham (2014)
45. Volz, V., Rudolph, G., Naujoks, B.: Surrogate-assisted partial order-based evolutionary optimisation. In: Conference on Evolutionary Multi-Criterion Optimization (EMO), pp. 639–653. Springer, Cham, CH (2017)
46. Wang, H., van Stein, B., Emmerich, M., Bäck, T.: Time complexity reduction in efficient global optimization using cluster kriging. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '17, pp. 889–896. ACM, New York, NY, USA (2017)
47. Weihs, C.: Moi-mbo: Multiobjective infill for parallel model-based optimization. In: Learning and Intelligent Optimization: 8th International Conference, Lion 8, Gainesville, FL, USA, February 16-21, 2014. Revised Selected Papers, vol. 8426, p. 173. Springer (2014)
48. Wessing, S., Preuss, M.: The true destination of EGO is multi-local optimization. ArXiv e-prints, arXiv:1704.05724 (2017)
49. Yao, X.: An empirical study of genetic operators in genetic algorithms. Microprocessing and Microprogramming **38**(1), 707 – 714 (1993)
50. Zaefferer, M., Stork, J., Friese, M., Fischbach, A., Naujoks, B., Bartz-Beielstein, T.: Efficient global optimization for combinatorial problems. In: Proceedings of the 2014 Conference on Genetic and Evolutionary Computation, GECCO '14, pp. 871–878. ACM, New York, NY, USA (2014)